Combination of nonconforming finite element method ϕ-FEM with neural networks

Vanessa LLERAS

University of Montpellier

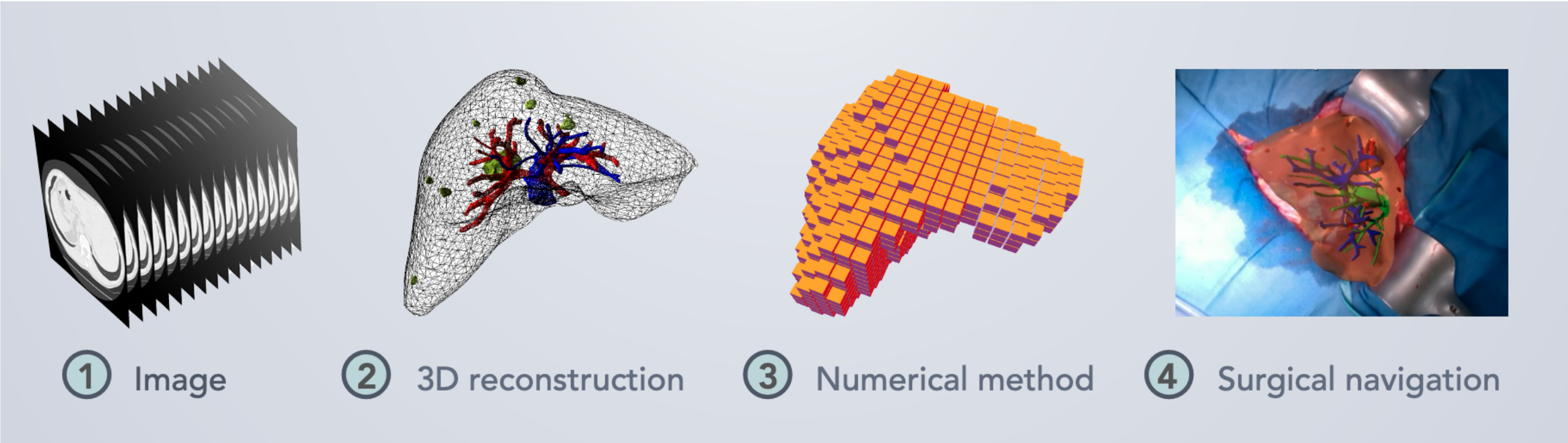
Institut Montpelliérain Alexander Grothendieck

Joint work with Michel Duprez (INRIA Mimesis),
Alexei Lozinski (University of Franche-Comté), Vincent Vigon (University of Strasbourg)
and Killian Vuillemot (University of Montpellier)

Outline

Motivation

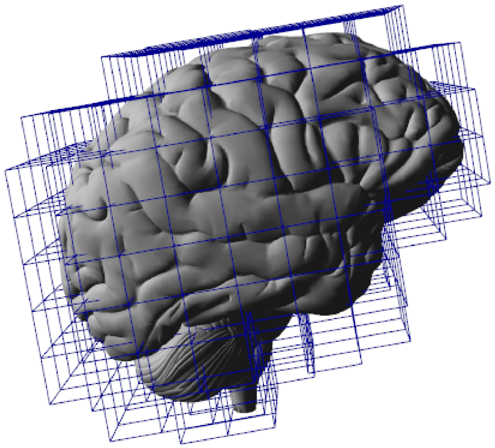
Objectives : develop real-time, predictive digital twins for computer-aided interventions in the fields of surgery, interventional radiology, and neuro-stimulation.



- What can we do on complex geometries ?
How can we simulate the deformation of soft tissues ?

Motivation

- Solving PDEs using finite element methods on non matching grids



— A simpler treatment of complex geometries, cracks, material interfaces, ...
— we can treat domain changing on iterations : Inverse problems, shape optimization
— we can treat domain changing in time : Fluid-Structure interaction, particulate flows, ...

⊕ No need to remesh,
⊕ regular cells to facilitate an efficient matrix-free implementation

⊖ adapt the weak formulation
⊖ Conditioning of the finite element matrix

Motivation

- Combining machine learning with numerical methods

Conventional methods

— solve one instance
— require the explicit form
— trade off on resolution
— slow on fine grids, fast on coarse grids

Data driven methods

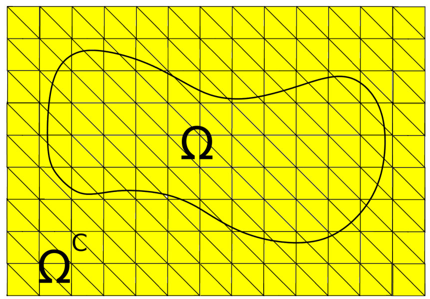
— Learn a family of PDE
— data driven
— resolution invariant, mesh invariant
— slow to train, fast to evaluate

Immersed boundary/ unfitted mesh methods may be useful in Deep Learning applications
A simple representation of the geometry is desirable if one want to learn the map

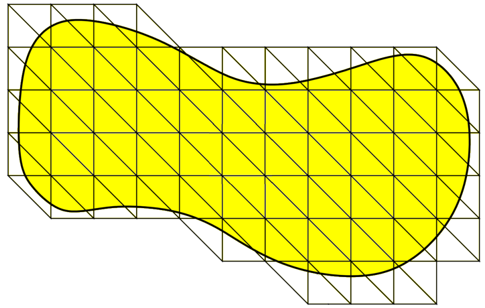
$$\text{(geometry of domain)} \rightarrow \text{(solution on domain)}$$

Previous works on non matching grids

- **Classical fictitious domain methods** *Saul'ev '63, Astrakhantsev '78, Glowinski et al. 1990's*
  ⊕ Easy to implement
  ⊖ poor accuracy $O(\sqrt{h})$
  ⊖ large FE matrix and bad condition number

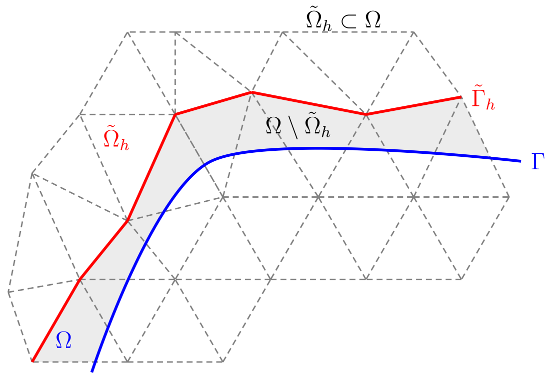


- **XFEM** *Moes-Bechet-Tourbier '06, Haslinger-Renard '09*
  ⊕ Good condition number
  ⊖ Non-classical shape functions and discontinuity in the integrals



- **CutFEM** *Burman-Hansbo 2010-2014*
  ⊕ Optimal accuracy
  ⊖ Not straigtforward to implement : cut integrals

- **Shifted Boundary Method (SBM)** : *Main-Scovazzi '17, Nouveau and al.*
  - Taylor development near the boundary
  ⊕ Optimal accuracy, no integrals on cut elements
  ⊖ Treatment of Neumann conditions
  ⊖ Require more geometrical information

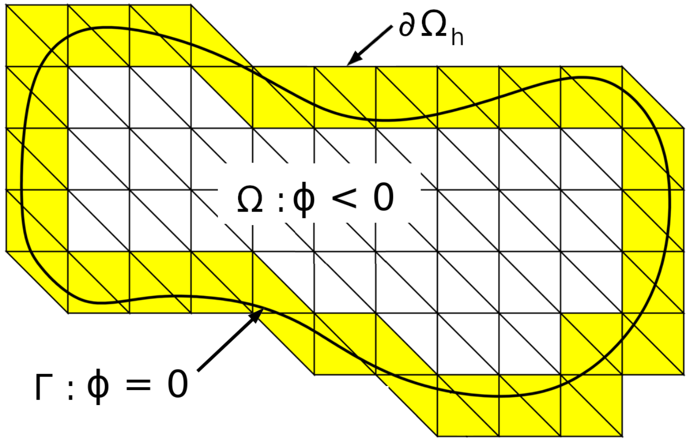


What is the idea of ϕ-FEM ?

Let the domain Ω and its boundary Γ be given by a level-set function ϕ :

$$\Omega := \{\phi < 0\} \text{ and } \Gamma = \{\phi = 0\}$$

Ω_h only slightly larger than Ω.



$\mathcal{T}_h$: ϕ-FEM mesh
$\mathcal{T}_h^\Gamma$: Cells of $\mathcal{T}_h$ cut by the boundary
$\mathcal{F}_h^\Gamma$: Internal facets of $\mathcal{T}_h^\Gamma$

What is the idea of ϕ-FEM ?

General procedure : $-\Delta u = f$ in Ω, $u = 0$ on $\partial\Omega$

— Extend the governing equations from Ω to Ω_h and write down a non standard variational formulation on the extended domain Ω_h without taking into account the boundary conditions on $\partial\Omega$.

$$\int_{\Omega_h} \nabla u \cdot \nabla v - \int_{\partial\Omega_h} (\partial_n u) v = \int_{\Omega_h} f v$$

— Impose the boundary conditions using appropriate ansatz or additional variables, explicitly involving the level set ϕ :

$$\boldsymbol{u} = \phi \boldsymbol{w}$$

— Add appropriate stabilization, including the ghost penalty as in CutFEM plus a least square imposition of the governing equation on the mesh cells near the boundary, to guarantee coerciveness/stability on the discrete level.

— The level set is known only approximately, ϕ_h is the Lagrange interpolation of ϕ of order $l \geq k+1$

— Find w_h (FEM of degree k) such that

$$\int_{\Omega_h} \nabla(\phi_h w_h) \cdot \nabla(\phi_h z_h) - \int_{\partial\Omega_h} (\partial_n \phi_h w_h) \phi_h z_h + Stab.terms = \int_{\Omega_h} f \phi_h z_h + Stab.terms$$

What is the idea of ϕ-FEM ?

General procedure : $-\Delta u = f$ in Ω, $u = 0$ on $\partial\Omega$

- Extend the governing equations from Ω to Ω_h and write down a non standard variational formulation on the extended domain Ω_h without taking into account the boundary conditions on $\partial\Omega$.

$$\int_{\Omega_h} \nabla u \cdot \nabla v - \int_{\partial\Omega_h} (\partial_n u) v = \int_{\Omega_h} f v$$

- Impose the boundary conditions using appropriate ansatz or additional variables, explicitly involving the level set ϕ :

$$\boldsymbol{u} = \phi \boldsymbol{w}$$

- Add appropriate stabilization, including the ghost penalty as in CutFEM plus a least square imposition of the governing equation on the mesh cells near the boundary, to guarantee coerciveness/stability on the discrete level.
- The level set is known only approximately, ϕ_h is the Lagrange interpolation of ϕ of order $l \geq k+1$
- Find w_h (FEM of degree k) such that

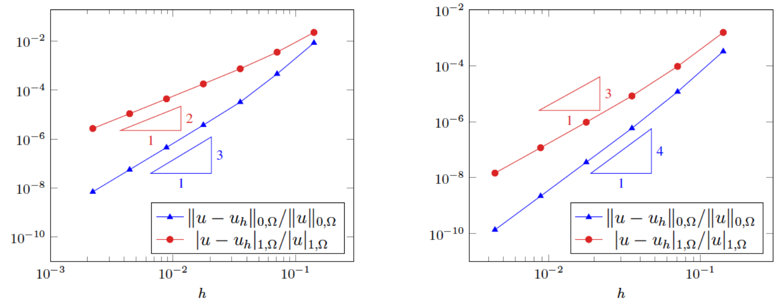
$$\begin{aligned}
&\int_{\Omega_h} \nabla(\phi_h w_h) \cdot \nabla(\phi_h z_h) - \int_{\partial\Omega_h} (\partial_n \phi_h w_h) \phi_h z_h \\
&+ \sigma_D h \sum_{E \in \mathcal{F}_h^\Gamma} \int_E [\partial_n(\phi_h w_h)] [\partial_n(\phi_h z_h)] + \sigma_D h^2 \sum_{T \in \mathcal{T}_h^\Gamma} \int_T \Delta(\phi_h w_h) \Delta(\phi_h z_h) \\
&= \int_{\Omega_h} f \phi_h z_h + -\sigma_D h^2 \sum_{T \in \mathcal{T}_h^\Gamma} \int_T f \Delta(\phi_h z_h)
\end{aligned}$$

What is the idea of ϕ-FEM ?

• easy of implementation : standard shape functions, all the integrals can be computed by standard quadrature rules on entire mesh cells and on entire boundary facets.

• Optimal convergence : in the L^2 norm : sub-optimal in theory, optimal in practice.

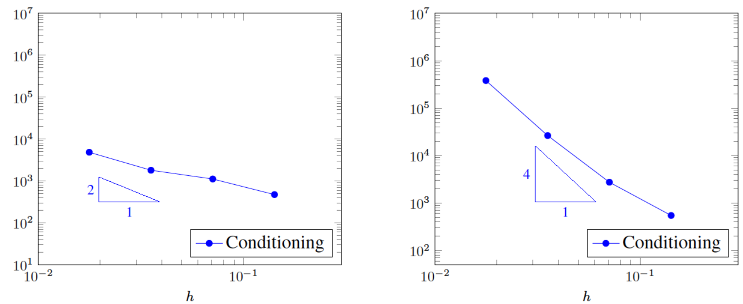
• ϕ-FEM works high polynomial orders : it suffices to approximate the level set function by piecewise polynomials of the same degree as that used for the primal unknown.



P_2 finite elements ; P_3 finite elements

• Good conditioning of the matrix : The finite element matrix of ϕ-FEM satisfies

$$\kappa(A) := \|A\|_2 \|A^{-1}\|_2 \leq Ch^{-2}$$



With stabilisation. Without stabilisation

• The method works for elasticity problem, a simple fracture problem, Stokes problem and an example of particulate flows, heat equation.

Neural Networks : choice of FNO

- FNO uses FFT, so that the solution should be represented on a Cartesian grid

- accurate than other deep learning method, faster than conventional solvers :

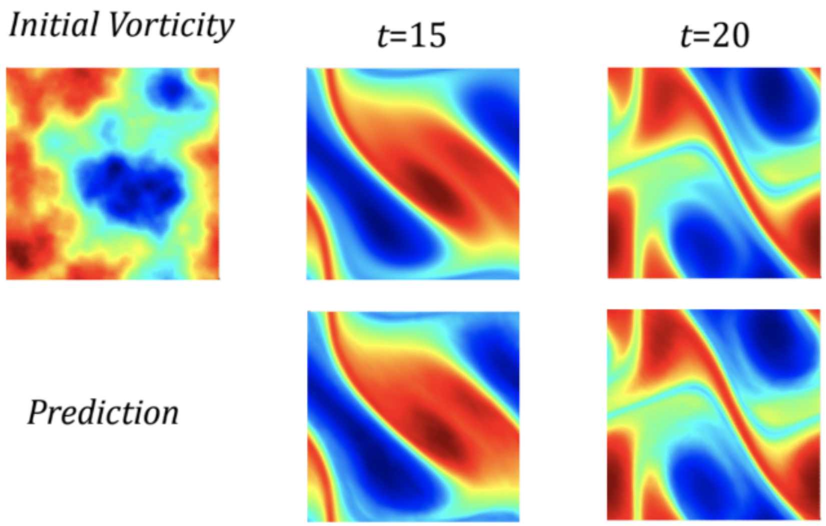
In *KOVACHKI et al, Neural Operator : Learning Maps Between Function Spaces (2022)*, the mean relative L2 errors on meshes $N \times N$

Networks	$N = 85$	$N = 141$	$N = 211$	$N = 421$
NN	0.1716	0.1716	0.1716	0.1716
FCN	0.0253	0.0493	0.0727	0.1097
PCANN	0.0299	0.0298	0.0298	0.0299
RBM	0.0244	0.0251	0.0255	0.0259
DeepONet	0.0476	0.0479	0.0462	0.0487
GNO	0.0346	0.0332	0.0342	0.0369
LNO	0.0520	0.0461	0.0445	–
MGNO	0.0416	0.0428	0.0428	0.0420
FNO	**0.0108**	**0.0109**	**0.0109**	**0.0098**

- it takes a step size much bigger than is allowed in numerical methods

Neural Networks : choice of FNO

- FNO demonstrate very good efficiency for different settings : for example Navier Stokes :



- the network can perform multi-resolution
- the training can be made on many PDEs with the same underlying architecture.

Neural Networks : choice of FNO

The FNO is a parametric application :

$$\mathcal{G}_\theta^\dagger : \mathbb{R}^{ni\times nj\times 3} \xrightarrow{P} \mathbb{R}^{ni\times nj\times nk} \longrightarrow \mathbb{R}^{ni\times nj\times nk} \xrightarrow{Q} \mathbb{R}^{ni\times nj\times 1}$$

ni is the number of pixels in the height, nj in the width.
Our FNO is composed of 4 layers with the same structure :

$$\mathcal{G}_\theta = \mathcal{H}_\theta^4 \circ \mathcal{H}_\theta^3 \circ \mathcal{H}_\theta^2 \circ \mathcal{H}_\theta^1$$

A layer is made of two sub-layers organised as followed :

$$\mathcal{H}_\theta^\ell(X) = \sigma\big(\mathcal{F}^{-1}\Big(\mathcal{F}(X)\cdot R\Big) + \mathcal{W}(X)\big)$$

where

— σ is an activation function applied term by term on the tensors. For $\ell = 1, 2, 3$ we choose the Relu function ($f(x) = \max(0, x)$) . For the last layer $\ell = 4$ we choose the GeLu function $f(x) = x\Phi(x)$ with $\Phi(x) = P(X \leqslant x)$ where $X \sim \mathcal{N}(0, 1)$

— $\mathcal{W}$ is the bias-layer.

Neural Networks : choice of FNO

— $\mathcal{F}$ the 2 dimensional Discrete Fourier transform (DFT) on the $ni \times nj$ grid :

$$\mathcal{F}(X)_{ijk} = \sum_{i'j'} X_{i'j'k} e^{2\sqrt{-1}\pi \frac{ii'}{ni}\frac{jj'}{nj}}$$

and its inverse :

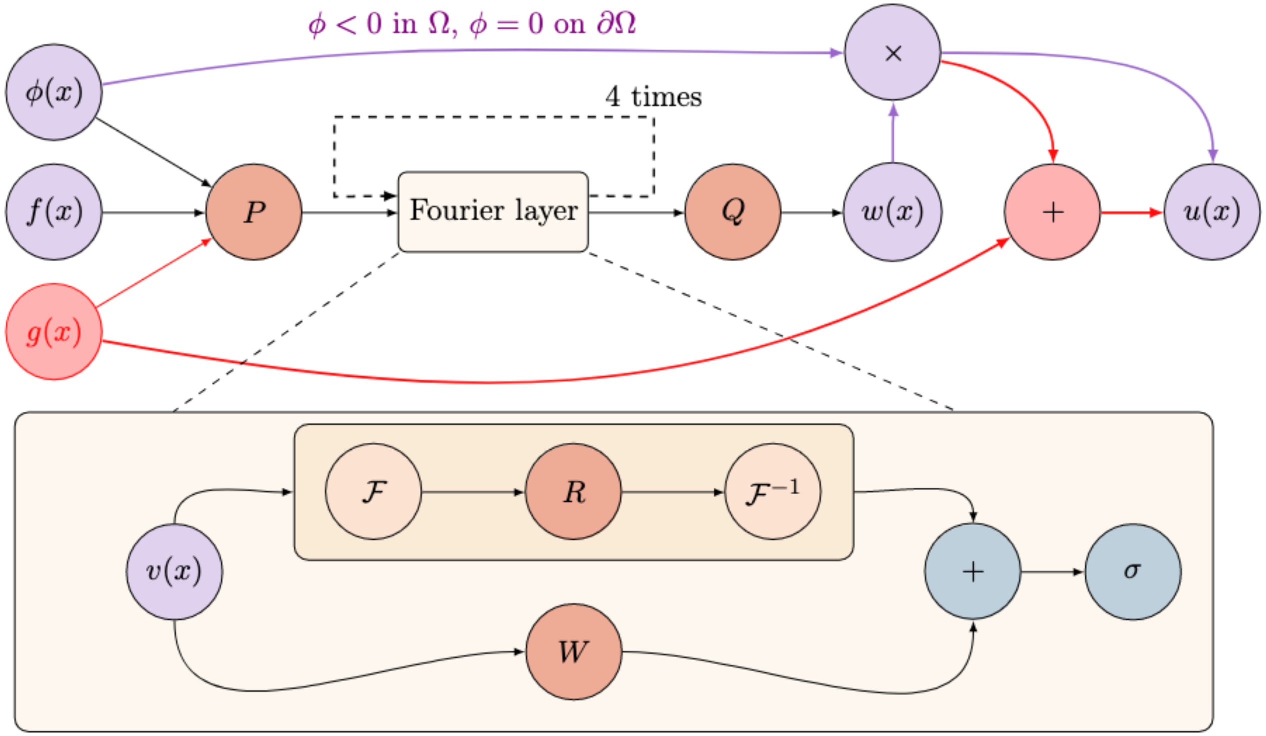
$$\mathcal{F}^{-1}(X)_{ijk} = \frac{1}{ni}\frac{1}{nj} \sum_{i'j'} X_{i'j'k} e^{-2\sqrt{-1}\pi \frac{ii'}{ni}\frac{jj'}{nj}}$$

— for our filtering task, it is sufficient to act only on the "low" frequencies. The multiplication $\mathcal{F}(X) \cdot R$ must simply be performed on the indices in the domain :

$$[0, mi[\times [0, mj[\quad \bigcup \quad [ni, ni - mi[\times [0, mj[$$

with $mi < ni/2$ and $mj < nj/2$. These parameters mi, mj are called : the number of Fourier modes of the filtering. Here $mi = mj = 20$.

Neural Networks : choice of FNO



$X = (f, \phi, f_x, f_y, f_{xx}, f_{yy}, \text{domain})$
P lifts the input to a high dimensional channel space
Q projects the representation back to the other space
R : Linear transformation applied on lower Fourier modes
W : Linear transformation applied on the spatial domain
σ : Activation function

Neural Networks

— Poisson-Dirichlet on different domains :

$$\begin{cases} -\Delta u &= f\,, \quad \text{in } \Omega\,, \\ u &= 0\,, \quad \text{on } \Gamma\,, \end{cases}$$

Goal : Learn the operator mapping the force and the level-set function to the solution,

$$\mathcal{G}^\dagger : (f, \phi) \mapsto w$$

$$\phi_{(x_0,y_0,l_x,l_y,\theta)}(x,y) = -1 + \frac{((x-x_0)\cos(\theta) + (y-y_0)\sin(\theta))^2}{l_x^2} + \frac{((x-x_0)\sin(\theta) - (y-y_0)\cos(\theta))^2}{l_y^2}\,,$$

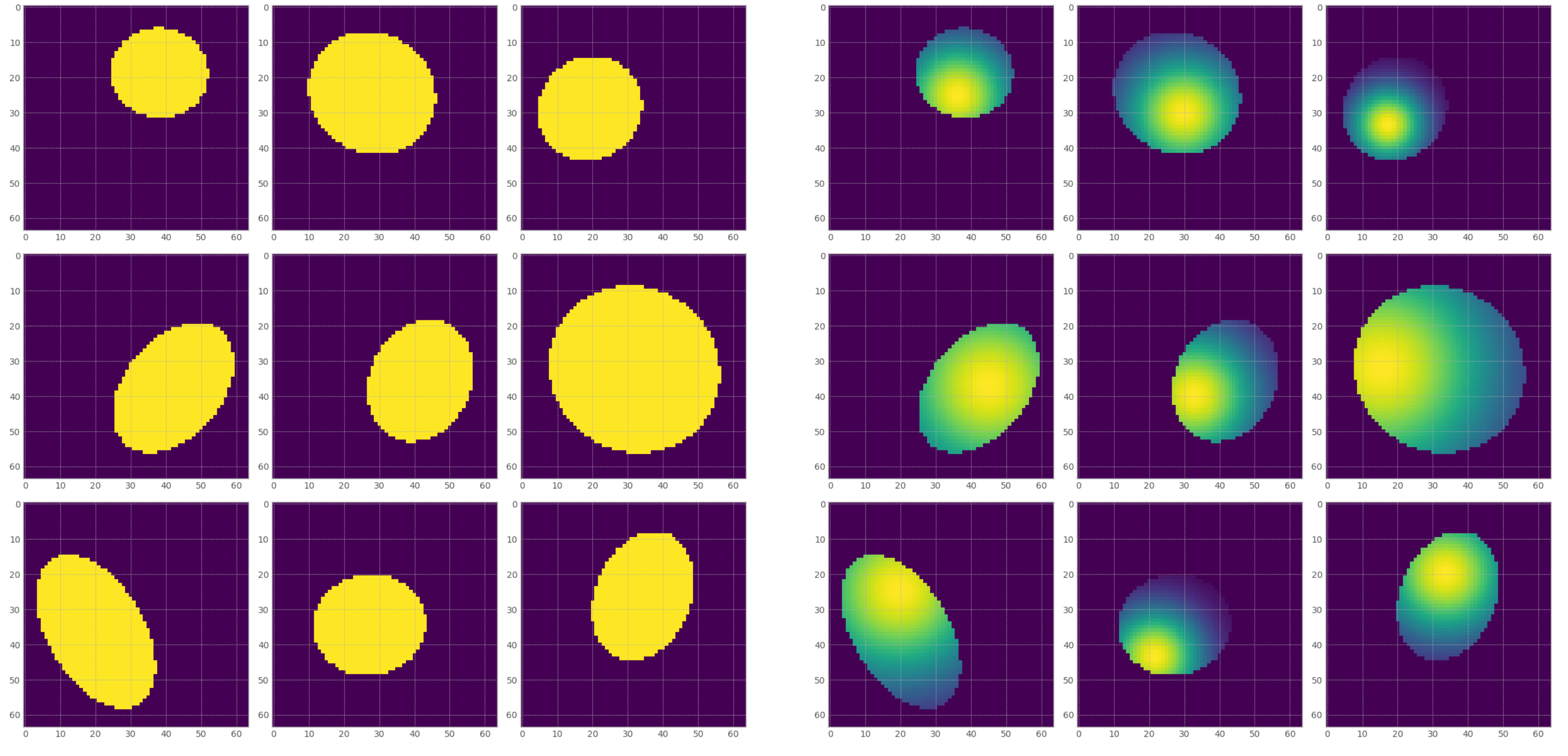
with

$$x_0,\ y_0 \sim \mathcal{U}([0.2, 0.8])\,, \quad l_x,\ l_y \sim \mathcal{U}([0.2, 0.45]) \quad \text{and } \theta \sim \mathcal{U}([0,\pi])\,.$$

$f = 100 \exp\left(-\frac{(x-\mu_0)^2+(y-\mu_1)^2}{2\sigma^2}\right)$, where μ_0 and μ_1 are chosen uniformly on $[0.2, 0.8]$ and $\sigma \sim [\min(l_x, l_y)2, \max(l_x, l_y)]$

Neural Networks

- 2000 epochs of training ; a batch of 64 samples is chosen on each epoch.
- Adam optimizer with an initial learning rate of 10^{-3},
- complete dataset of size 1500, divided in a training set of size 1313 and testing set of size 187



Neural Networks

loss function to learn : $loss = misfit_0 + misfit_1 + misfit_2$

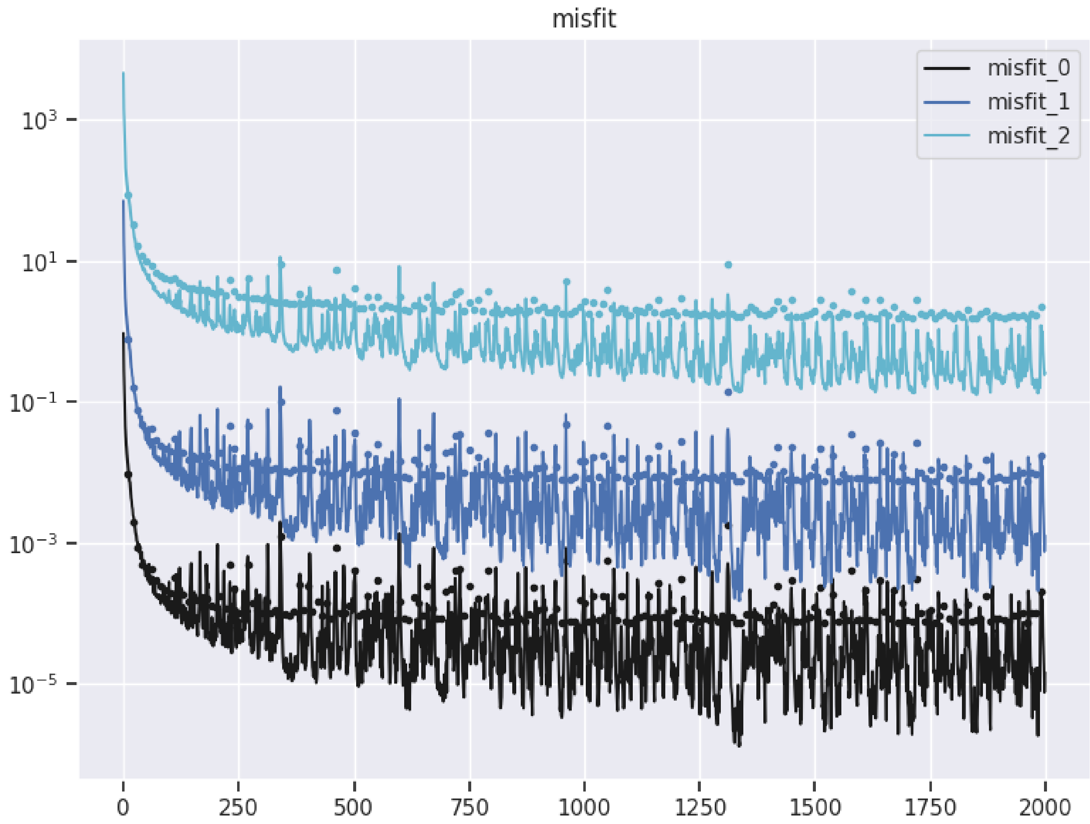
— Evolution of misfits on epochs of training :

$$\text{misfit}_0 = \frac{1}{N}\sum_{n=0}^{N} \frac{\|\phi_h^n \mathcal{G}_\theta^\dagger(\phi_h^n, f_h^n) - \phi_h^n w_h^n\|^2_{L^2(\Omega_h^n)}}{|\Omega_h^n|^2},$$

$$\text{misfit}_1 = \frac{1}{N}\sum_{n=0}^{N} \frac{\|\nabla_x(\phi_h^n \mathcal{G}_\theta^\dagger(\phi_h^n, f_h^n)) - \nabla_x(\phi_h^n w_h^n)\|^2_{L^2(\Omega_h^n)}}{|\Omega_h^n|^2}$$
$$+ \frac{\|\nabla_y(\phi_h^n \mathcal{G}_\theta^\dagger(\phi_h^n, f_h^n)) - \nabla_y(\phi_h^n w_h^n)\|^2_{L^2(\Omega_h^n)}}{|\Omega_h^n|^2},$$

$$\text{misfit}_2 = \frac{1}{N}\sum_{n=0}^{N} \frac{\|\nabla_x\nabla_x(\phi_h^n \mathcal{G}_\theta^\dagger(\phi_h^n, f_h^n)) - \nabla_x\nabla_x(\phi_h^n w_h^n)\|^2_{L^2(\Omega_h^n)}}{|\Omega_h^n|^2}$$
$$+ \frac{\|\nabla_y\nabla_y(\phi_h^n \mathcal{G}_\theta^\dagger(\phi_h^n, f_h^n)) - \nabla_y\nabla_y(\phi_h^n w_h^n)\|^2_{L^2(\Omega_h^n)}}{|\Omega_h^n|^2}$$
$$+ \frac{\|\nabla_x\nabla_y(\phi_h^n \mathcal{G}_\theta^\dagger(\phi_h^n, f_h^n)) - \nabla_x\nabla_y(\phi_h^n w_h^n)\|^2_{L^2(\Omega_h^n)}}{|\Omega_h^n|^2},$$

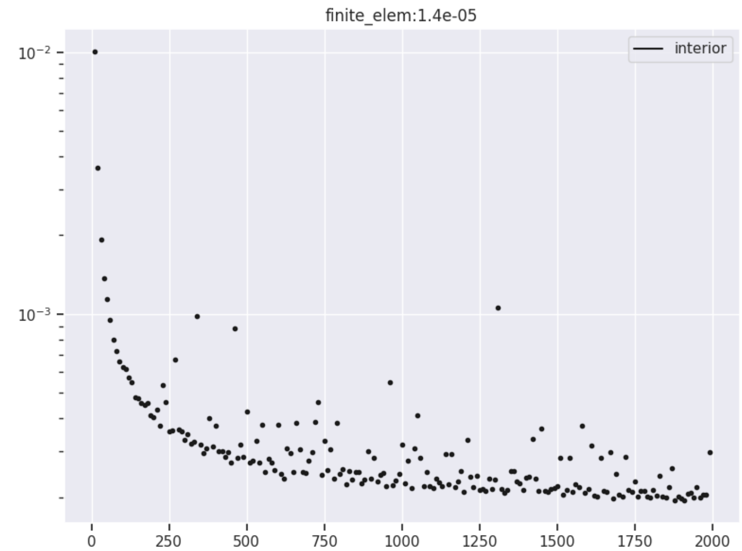
Neural Networks



Neural Networks

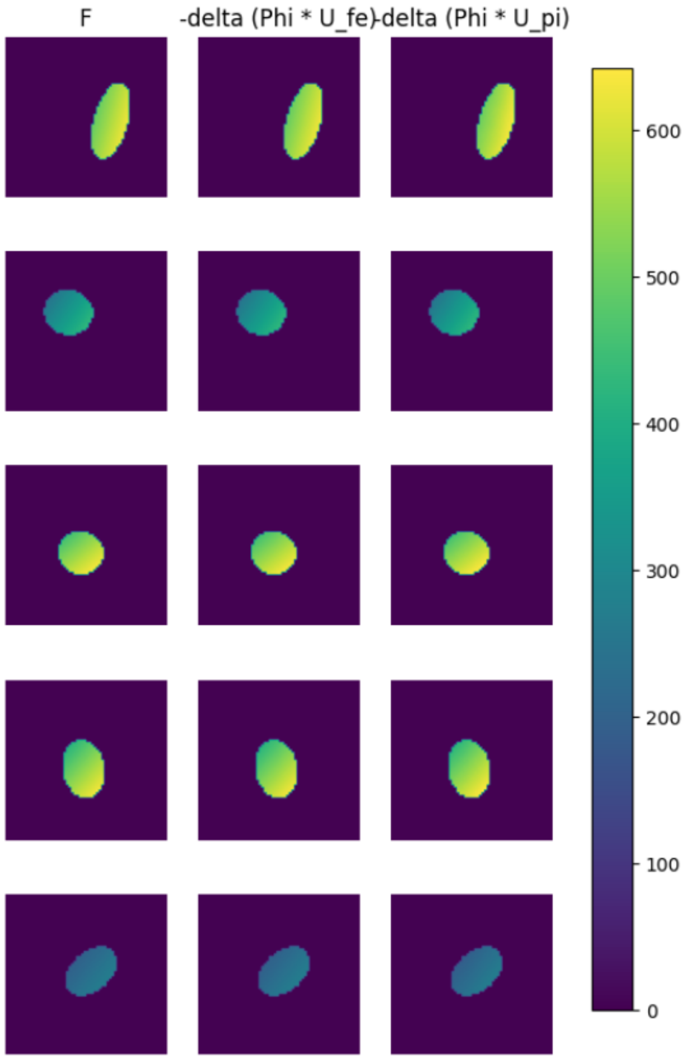
interior relative residues, given by

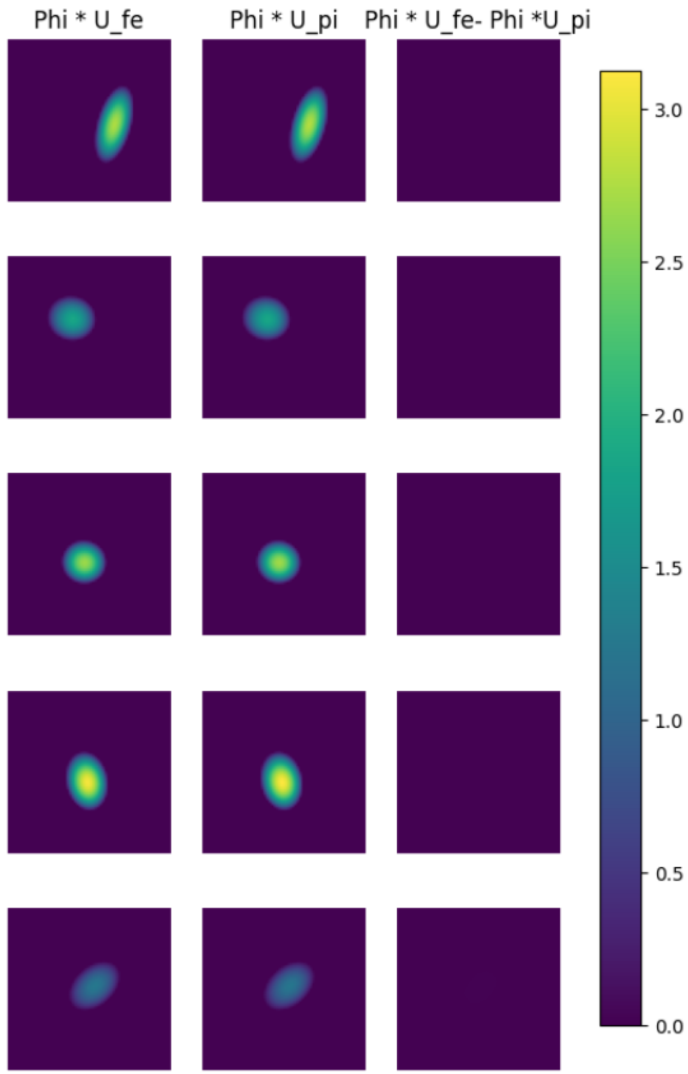
$$r_\theta = \frac{1}{N}\sum_{n=1}^{N} \frac{\|(\Delta(\phi_h^n \mathcal{G}_\theta^\dagger(\phi_h^n, f_h^n)) + f_h^n)/|\Omega_h^n|\|_{L^2(\Omega_h^n)}^2}{\|f_h^n/|\Omega_h^n|\|_{L^2(\Omega_h^n)}^2}.$$



the residues seems to converge to $\approx 2 \times 10^{-4}$ on the validation set, whereas the ϕ-FEM residues on the same sample are 1.4×10^{-5}

Neural Networks

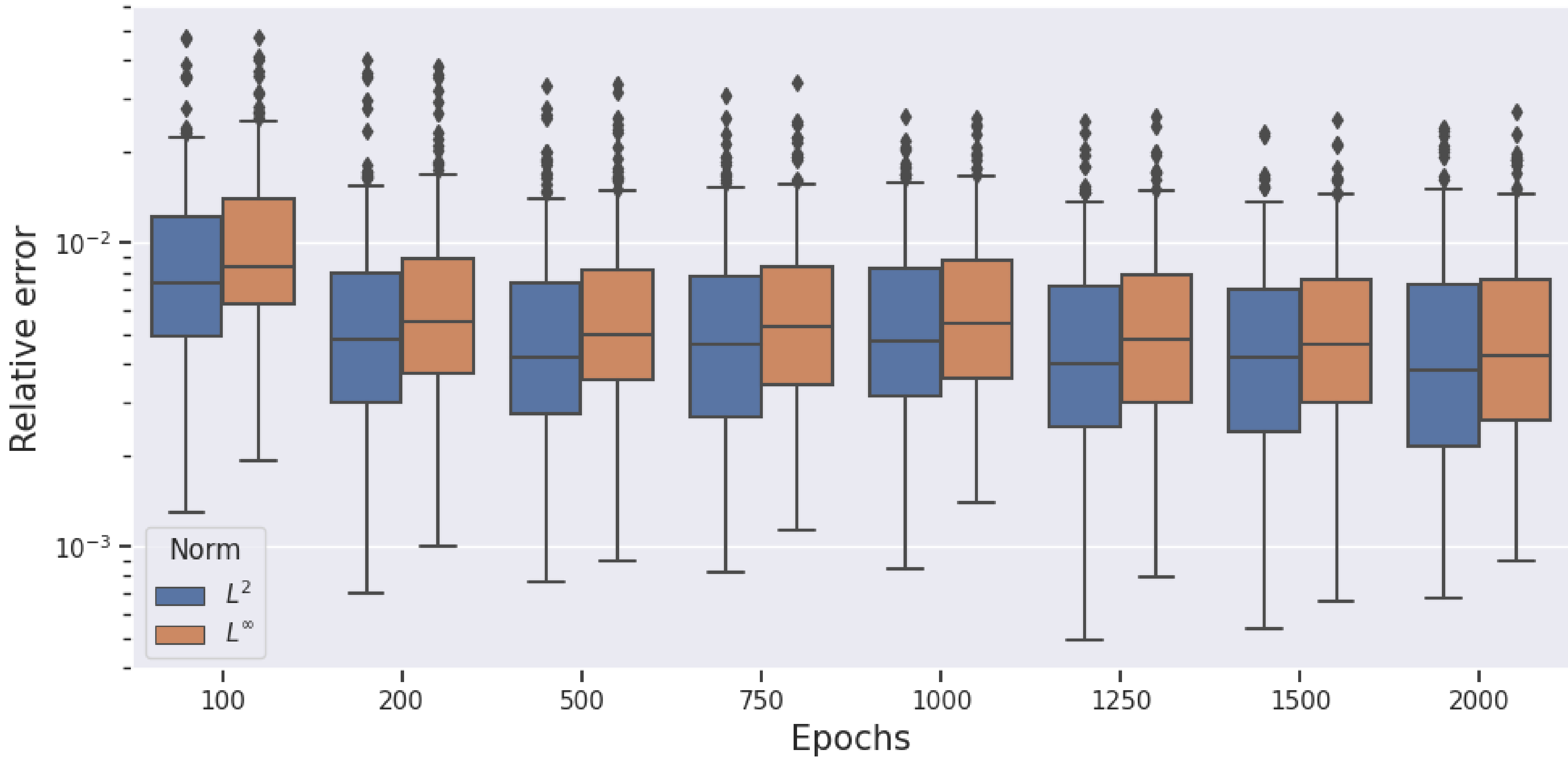




Neural Networks

Error introduced by the FNO : Let us look at :

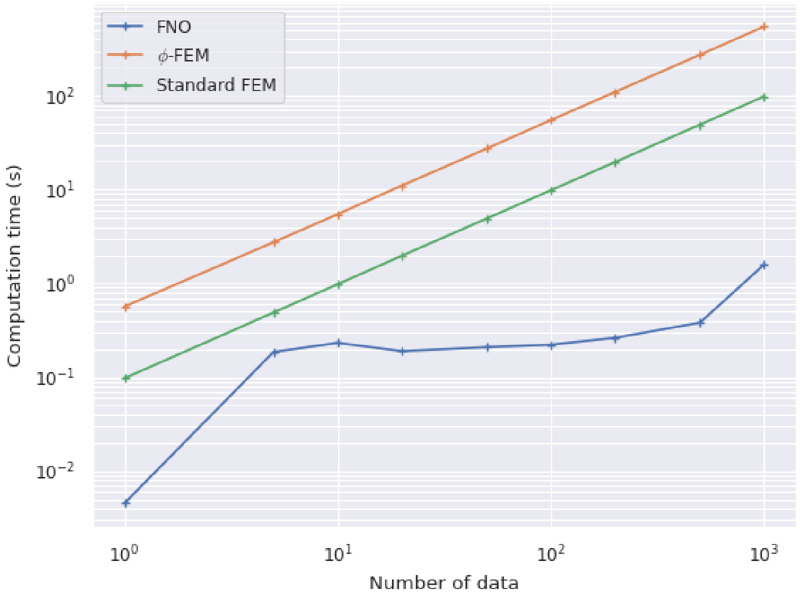
$$\frac{\|\phi_h w_h - \phi_h \mathcal{G}_\theta^\dagger(\phi_h, f_h)\|^2_{0,\Omega_h}}{\|\phi U_{ref}\|^2_{0,\Omega_h}}$$



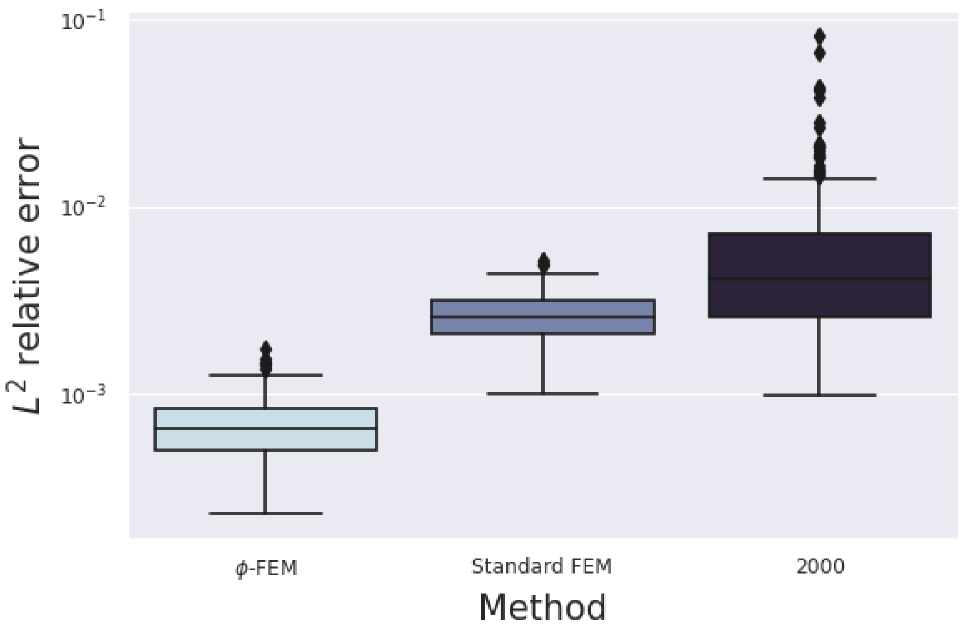
Relative L^∞ and L^2 errors on the validation set, at different steps of the training.

Neural Networks

Computation times



Relative error with different methods



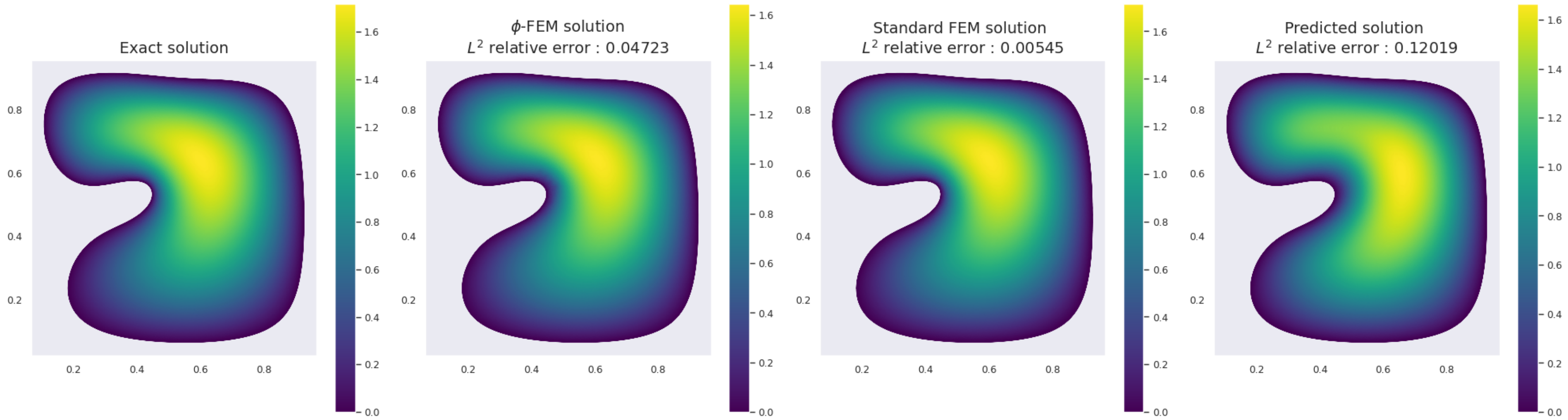
Neural Networks

Case of non-parametric domains :

Let us denote Ω the constructed domain and $\tilde{\phi}$ the created function with Fourier series. The level-set ϕ is given by

$$\phi = -\left(\tilde{\phi} - \min_{\Omega}(\tilde{\phi})\right)|_{\Omega} + \left(\left||\tilde{\phi}| - \min_{\Omega}(\tilde{\phi})\right|\right)|_{\Omega^c},$$

where Ω^c is the complement of Ω in $(0,1)^2$.



Conclusion and ongoing works

Results :

- ϕ-FEM has several attractive features :
  Optimal convergence, discrete problem well conditioned, simple implementation, formulation available for any order of approximation, ϕ-FEM works for several problems.
- Training neural operators could be expensive, but we have shown that after training, the FNO compute faster than finite element methods or phifem method.

Future works :

- ϕ-fem and finite differences
- Comparison with ϕ-fem approach combined with CNN